

CÓDIGOS BINARIOS

SISTEMAS DE CODIFICACIÓN

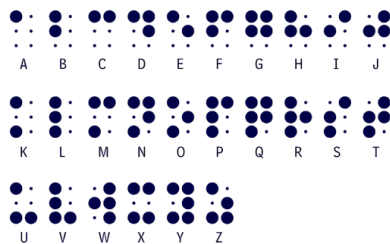
Edición: 091113

Luis González
Profesor de Tecnologías de la Información
Departamento de Tecnología
I.E.S. Santa Eugenia

DATOS E INFORMACIÓN

Conviene diferenciar el significado de los términos **datos** e **información**. La **información** es un concepto muy amplio, que engloba **todas las posibles formas del conocimiento humano**. Todos los fenómenos que percibimos, ya sea con nuestros sentidos o mediante instrumentos, pueden contener información. Dicha información sólo será inteligible para aquellas personas que conozcan su significado, es decir, para aquellas personas que pueden asociar el significante y su contenido.

Los **datos**, en cambio, son fragmentos de información codificada, lista para ser interpretada y procesada, ya sea por una máquina o por un ser humano. Los **datos**, como tales, carecen de **significado**, y solo lo alcanzan cuando son **interpretados**; una vez que los datos han sido procesados y se muestra su resultado **de modo inteligible**, pasan a formar parte del flujo de información.



Por ejemplo, una secuencia de puntos, que sobresalen de una superficie siguiendo un patrón geométrico, no pueden tener un origen natural, fruto del azar. Es obvio que contienen información, pero ¿qué significan? Sólo una persona que conozca la equivalencia, entre cada patrón de puntos y un carácter alfanumérico, podrá acceder a su significado. A la izquierda tienes la equivalencia entre puntos y caracteres del código Braille, el que se utiliza para escribir textos para personas invidentes.

La ventaja del uso de ordenadores estriba en que son capaces de procesar gigantescas cantidades de datos en muy poco tiempo. Los ordenadores pueden almacenar y manipular datos, pero no pueden interpretarlos. Debe instalarse en ellos los programas capaces de interpretar y procesar los datos: lectores de archivos, *codecs* de audio y video, etc.

CODIFICACIÓN BINARIA

Para que los ordenadores puedan manipular datos, deben recibirlos **codificados**. Aunque pueden utilizarse códigos muy diversos, todos los códigos empleados en computación tienen una característica común: sólo utilizan dos signos, los dígitos 0 y 1.

La razón de utilizar sólo dos dígitos se debe a que todos los dispositivos de un ordenador (el procesador, la memoria, etc.) están contruidos con circuitos electrónicos basados en transistores, que sólo utilizan dos estados¹: tensión alta o tensión baja, circuito abierto o circuito cerrado, pasa corriente o no pasa corriente, etc. Asociamos esos estados con los dígitos 1 y 0 y eso nos permite codificar la información.

1. CÓDIGO BINARIO PURO

La codificación binaria está basada en el sistema de numeración binario, que utiliza los dígitos 0 y 1 para representar cualquier número. El binario es, como los demás sistemas de numeración que utilizamos, un sistema posicional, en el que cada dígito tiene un peso que depende de su posición en la cifra.

El peso que tiene cada posición de la cifra es el siguiente:

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
16	8	4	2	1	0,5	0,25	0,125

Así, por ejemplo, para convertir a decimal el número binario entero 10011 basta con tener en cuenta el peso de cada dígito y sumarlos:

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	0	0	1	1	0	0	0
16	8	4	2	1	0,5	0,25	0,125

$$16 + 2 + 1 = 19$$

Y lo expresamos así: $10011_2 = 19_{10}$

indicando con los subíndices 2 y 10 la base de numeración en la que están expresados.

¹ Los transistores que componen los circuitos digitales trabajan tan sólo en dos estados: corte (la corriente de colector es nula y la tensión del colector es la misma que la alimentación, unos 5 voltios en tecnología TTL) y saturación (la corriente del colector alcanza el máximo valor posible y la tensión del colector es prácticamente 0 voltios). Así pues, el colector de los transistores sólo conoce dos estados: 5 voltios y 0 voltios.

Pongamos un ejemplo de conversión del número binario 11001.011 Como ves, no es un número entero sino que tiene decimales separados por un punto de la parte entera. Lo haremos del mismo modo que en el ejemplo anterior, asignado a cada dígito el peso que le corresponde por su posición:

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	1	0	0	1	0	1	1
16	8	4	2	1	0,5	0,25	0,125
$16 + 8 + 1 + 0,25 + 0,125 = 25,375$							

Y lo expresamos así: $11001.011_2 = 25,375_{10}$

2. CÓDIGO HEXADECIMAL

Cuando expresamos un dato en código binario suelen formarse largas palabras de ceros y unos, difíciles de retener e interpretar a simple vista. Observa el siguiente dato, ¿te parece fácil de recordar?:

110101000100101010101101110101110010110111001

El código hexadecimal es un modo de **presentar** los datos binarios, de tal modo que la longitud de las palabras se reduce a la cuarta parte. El dato binario anterior, expresado en hexadecimal, queda así:

3512ab7ae5b9

La longitud de las palabras binarias no asusta a las máquinas, las procesan ciegamente. Pero programamos el software para que, al presentar datos a un usuario humano lo haga en forma hexadecimal, que es mucho más fácil de leer. Por ejemplo:

un color rgb **2f3ad7**
 una dirección IP **2e.f4.c5.07**
 una dirección de memoria **0x00f3061a**

Cuando el ordenador presenta números en formato hexadecimal en la pantalla, suele indicarlo así, anteponiendo un cero y una x (0x) al número para indicar que está expresado en hexadecimal.

La relación entre un código binario y su expresión hexadecimal es muy simple. Tomando los dígitos binarios en grupos de cuatro, los sustituimos por su equivalente hexadecimal a partir de la siguiente tabla de equivalencias:

<i>bin</i>	<i>hex</i>	<i>bin</i>	<i>hex</i>	<i>bin</i>	<i>hex</i>	<i>bin</i>	<i>hex</i>
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

CONVERSIÓN BINARIO → HEXADECIMAL

Por ejemplo, para expresar en hexadecimal el siguiente dato binario:

101110001111011

hacemos grupos de cuatro bits, empezando desde la derecha, y completamos con ceros el último grupo si fuera necesario:

0101 1100 0111 1011
5 c 7 b

y sustituimos cada grupo de bits por su dígito hexadecimal equivalente:

$$101110001111011_2 = 0x5c7b$$

CONVERSIÓN HEXADECIMAL → BINARIO

La conversión de un número hexadecimal a su equivalente en binario es igualmente sencilla: cada cifra del número hexadecimal se sustituye por los cuatro bits que le corresponden en binario. Si en el número resultante hubiese ceros a la izquierda, los desecharemos porque son irrelevantes para el valor del número.

Por ejemplo, vamos a convertir al sistema binario el siguiente número hexadecimal:

$$6b1f3c$$

Cada una de los dígitos tiene la siguiente equivalencia:

$$\begin{array}{cccccc} 6 & b & 1 & f & 3 & c \\ 0110 & 1011 & 0001 & 1111 & 0011 & 1100 \end{array}$$

Así pues, podemos decir que:

$$0x6b1f3c = 11010110001111100111100_2$$

3. CÓDIGO BCD

El código binario puro resulta poco intuitivo para los que estamos habituados a manejar el sistema decimal, en el que las cifras se componen de unidades, decenas, centenas, etc. El código **BCD**² trata de ayudar a hacer el código binario más fácil de leer.

El modo de hacerlo es simple, **cada una de las cifras** del número decimal se convierte en un código binario de 4 dígitos de acuerdo con la siguiente tabla de equivalencia:

<i>bin</i>	<i>BCD</i>	<i>bin</i>	<i>BCD</i>
0000	0	0101	5
0001	1	0110	6
0010	2	0111	7
0011	3	1000	8
0100	4	1001	9

CONVERSIÓN DECIMAL → BCD

Pongamos un ejemplo: ¿cómo se expresa un número decimal como el 359 en BCD? Cada dígito de la cifra digital se convierte en su equivalente de 4 bits en BCD, así:

$$3_{10} = 0011_{BCD} \quad 5_{10} = 0101_{BCD} \quad 9_{10} = 1001_{BCD}$$

$$359_{10} = 001101011001_{BCD}$$

² Binary Coded Decimal: Número decimal codificado en binario

Segundo ejemplo: vamos a convertir un número decimal fraccionario, como el 217,36 a BCD utilizando la misma técnica de sustituir cada dígito de la cifra decimal por su equivalente BCD:

$$\begin{aligned} 2_{10} &= 0010_{\text{BCD}} & 1_{10} &= 0001_{\text{BCD}} & 7_{10} &= 0111_{\text{BCD}} & 3_{10} &= 0011_{\text{BCD}} & 6_{10} &= 0110_{\text{BCD}} \\ 216.36_{10} &= 001000010111.00110110_{\text{BCD}} \end{aligned}$$

CONVERSIÓN BCD → DECIMAL

El proceso inverso es igualmente sencillo: sustituimos cada grupo de cuatro bits, empezando por la derecha, por el dígito decimal equivalente. Así, por ejemplo, para convertir el siguiente código BCD:

11010010001.01110010

hallamos la equivalencia de cada grupo de cuatro bits:

0110	1001	0001	0111	0010
6	9	1	7	2

Por lo tanto:

$$11010010001.01110010_{\text{BCD}} = 691.72_{10}$$

4. CÓDIGO DE GRAY

Tomemos dos números consecutivos expresados en código binario puro, el 0001 y el 0010 (los números 1 y 2 en el sistema decimal), por ejemplo. Al pasar de un número al siguiente dos bits deben cambiar simultáneamente. Esto supone un problema físico en dispositivos de entrada-salida³: ambos cambios deben ocurrir en el mismo instante. Si uno de los cambios ocurre una fracción de tiempo antes que el otro, durante un breve instante aparecería otro número en el dispositivo. Podría ocurrir, por ejemplo, que apareciese la secuencia 0001 - 0011 - 0010 o la secuencia 0001 - 0000 - 0010, de forma incontrolada. Si analizas la transición entre el número 0111 (7) y el 1000 (8) la situación es aún más incierta: hasta cuatro bits deberían cambiar de valor simultáneamente. En estas situaciones, el ordenador debe recurrir a otro tipo de codificación.

Una solución muy utilizada es recurrir al código de Gray, en el que entre un número y el siguiente **sólo hay un bit que cambia de valor**. El código de Gray no es posicional, puesto que a cada una de las posiciones de la cifra no se les asigna un peso específico. Por eso el código de Gray no se utiliza en operaciones aritméticas sino en el tráfico de datos en los dispositivos de entrada-salida. Veamos una lista de códigos de Gray:

<i>dec</i>	<i>bin</i>	<i>Gray</i>	<i>dec</i>	<i>bin</i>	<i>Gray</i>
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0101	0110	12	1100	1010
5	0110	0111	13	1101	1011
6	0111	0101	14	1110	1001
7	1000	0100	15	1111	1000

³ Un dato recibido en un bus, un dato enviado a la impresora, etc.

UNIDADES DE MEDIDA DE LA INFORMACIÓN

La información o el conocimiento se transmite enviando datos, desde el emisor hasta el receptor, a través de un canal de comunicación. Pero ¿cómo se mide la información? ¿cuánta información entregamos al enviar un archivo de datos por un canal de comunicaciones?

La cantidad más pequeña de información que podemos enviar, a través de un canal, es la que comunica al receptor un hecho o un suceso en el que **sólo hay dos alternativas** posibles⁴. Para informar de cual de las dos posibilidades ha ocurrido, bastaría con enviar **un único dígito binario**, es decir un 0 o un 1. A esta cantidad mínima de información se le denomina **bit**, abreviatura de la palabra inglesa *binary digit* y se toma como unidad básica de información,

Al conjunto de 8 bits se le denomina **byte**. Un byte puede tomar un valor cualquiera entre 00000000 y 11111111. El número total de combinaciones posibles es de 256. Con un byte, por tanto, podemos comunicar 256 hechos distintos: 256 tonos de música, 256 caracteres de escritura, 256 colores, etc.

Asimismo, un byte está compuesto por dos **nibble**. Cada **nibble** está compuesto por cuatro bit y se puede representar por un carácter hexadecimal. Por ejemplo, el byte 10100111 está compuesto por dos **nibble**: 1010 (A) y 0111 (7). Ese byte se representa, en código hexadecimal así: 0xa7

Tanto el bit como el byte son unidades de medida muy pequeñas, por lo que se necesitan algunos múltiplos del byte. Así, hablamos de Kilobyte, Megabyte, Gigabyte, etc. En la tabla siguiente encontrarás la relación entre las distintas magnitudes:

1 byte	8 bits		
1 Kilobyte (KB)	1024 bytes	8192 bits	
1 Megabyte (MB)	1024 Kilobytes	$1,05 \cdot 10^6$ bytes	$8,39 \cdot 10^6$ bits
1 Gigabyte (GB)	1024 Megabytes	$1,05 \cdot 10^6$ Kilobytes	$1,07 \cdot 10^9$ bytes
1 Terabyte (TB)	1024 Gigabytes	$1,05 \cdot 10^6$ Megabytes	$1,1 \cdot 10^{12}$ bytes
1 Petabyte	1024 Tera	$1,05 \cdot 10^6$ Gigabytes	

El motivo de que la proporción entre las distintas magnitudes sea de 1024, en lugar de 1000 que es lo habitual en el sistema decimal, se debe a que 1024 es la potencia de base 2 que más se aproxima al múltiplo 1000 ($2^{10} = 1024$), equivalente al prefijo kilo en el sistema decimal.

⁴ Cierto o falso. Blanco o negro. Si o no. Abierto o cerrado, etc.

CÓDIGO ASCII

Como ya se ha indicado, el ordenador necesita tener los datos e instrucciones codificados en forma binaria, es decir, convertidos en 0 y 1; por tanto, **todos** los caracteres, las letras, los números y demás caracteres especiales del teclado, deben estar codificados mediante un código binario **unívoco**, es decir, que no pueda inducir a errores de interpretación.

Los **códigos de caracteres**, representan a cada uno de los caracteres disponibles en el teclado mediante números binarios, completándolos con ceros a la izquierda hasta formar **octetos** o **bytes** completos.

Existen distintos códigos de caracteres, pero el más utilizado sigue siendo el código **ASCII**⁵. En este sistema, a cada carácter le corresponde un número, que en el sistema decimal está comprendido entre **0** y **255** y, en el sistema hexadecimal, está comprendido entre el **00** y el **FF**. Cada carácter está representado, en el código **ASCII**, por un byte, es decir, por 8 bits:

Dec	Hex	Letra	Dec	Hex	Letra	Dec	Hex	Letra	Dec	Hex	Letra	Dec	Hex	Letra	Dec	Hex	Letra
0	0		22	16		44	2c	,	66	42	B	88	58	X	110	6e	n
1	1		23	17		45	2d	-	67	43	C	89	59	Y	111	6f	o
2	2		24	18		46	2e	.	68	44	D	90	5a	Z	112	70	p
3	3		25	19		47	2f	/	69	45	E	91	5b	[113	71	q
4	4		26	1a		48	30	0	70	46	F	92	5c	\	114	72	r
5	5		27	1b		49	31	1	71	47	G	93	5d]	115	73	s
6	6		28	1c		50	32	2	72	48	H	94	5e	^	116	74	t
7	7		29	1d		51	33	3	73	49	I	95	5f	_	117	75	u
8	8		30	1e		52	34	4	74	4a	J	96	60	`	118	76	v
9	9		31	1f		53	35	5	75	4b	K	97	61	a	119	77	w
10	a		32	20		54	36	6	76	4c	L	98	62	b	120	78	x
11	b		33	21	!	55	37	7	77	4d	M	99	63	c	121	79	y
12	c		34	22	"	56	38	8	78	4e	N	100	64	d	122	7a	z
13	d		35	23	#	57	39	9	79	4f	O	101	65	e	123	7b	{
14	e		36	24	\$	58	3a	:	80	50	P	102	66	f	124	7c	
15	f		37	25	%	59	3b	;	81	51	Q	103	67	g	125	7d	}
16	10		38	26	&	60	3c	<	82	52	R	104	68	h	126	7e	~
17	11		39	27	'	61	3d	=	83	53	S	105	69	i	127	7f	□
18	12		40	28	(62	3e	>	84	54	T	106	6a	j			
19	13		41	29)	63	3f	?	85	55	U	107	6b	k			
20	14		42	2a	*	64	40	@	86	56	V	108	6c	l			
21	15		43	2b	+	65	41	A	87	57	W	109	6d	m			

Para no confundirse, el ordenador agrupa los bits de cada letra en grupos completos de ocho, por lo que, si su código binario estuviera constituido por un número menor de dígitos, lo completaría añadiendo ceros a la izquierda. Por ejemplo, al teclear el carácter **C** (67₁₀) se introducirá y almacenará en su código binario 01000011 (0x43) o, al teclear la **barra espaciadora** (32₁₀), lo que se introduce en el registro del teclado es el código binario 00100000 (0x20).

Los 32 primeros caracteres del código ASCII están constituidos por los caracteres de control: **Intro**, **Delete**, etc. Los siguientes, hasta el 128, son caracteres internacionales y, por tanto, comunes para todos los países. Para obtener más información visita la siguiente web: <http://www.abcdatos.com/utiles/ascii.html>

Tras el éxito del primer PC, la tabla de caracteres ASCII se quedó pequeña. La pantalla sólo podía presentar números y letras. Por esa razón se diseñó el código ASCII extendido, que amplía los caracteres disponibles a signos gráficos, flechas, símbolos matemáticos, etc, y otros particulares de cada país, como, por ejemplo, nuestra característica **ñ**. Puedes ver el código ASCII extendido en esta web: <http://www.cdrummond.qc.ca/ce-gep/informat/Professeurs/Alain/files/ascii.htm>

Hoy, con la incorporación de países asiáticos y árabes a la Red y con las demandas de adaptación a su idioma por parte de países con lenguas vivas, pero de escasa implantación, el código ASCII ha sido desbordado. Un sistema internacional de codificación de caracteres, denominado **Unicode**, ha venido a proporcionar la solución para todos los signos y caracteres vigentes en el mundo. Si quieres saber más puedes consultar la **wikipedia** pinchando [en este enlace](#).

⁵ American Standard Code for Information Interchange

LAS INSTRUCCIONES DE LOS PROGRAMAS

También las instrucciones de los programas, que son órdenes que indican al procesador qué debe hacer con los datos, debe estar codificadas en binario. Cada procesador tiene un juego de instrucciones propio, que se ajusta a un código.

Por ejemplo, el procesador PIC 16C84, que es un pequeño procesador RISC, tiene un juego de 35 instrucciones de 14 bit de largo. A modo de ejemplo, te muestro en la tabla siguiente algunas de ellas:

ASSEMBLER	DESCRIPCIÓN	CÓDIGO
ADDWG g,d	Suma el contenido de W a G y lo envía al destino d	00 0111 dggg gggg
CLRG g	Pone a cero el registro G	00 0001 1ggg gggg
MOVWG g	Copia el contenido del registro W en G	00 0000 1ggg gggg
ADDLW K	Suma al registro W la constante K	11 111x kkkk kkkk

Si quieres consultar el resto de las instrucciones de este pequeño procesador, lo puedes hacer en la web del fabricante [Microchip WebSite](http://www.microchip.com)

LA IMPORTANCIA DE LAS NORMAS

Un verdadero torrente de datos circula por los canales o buses del ordenador. El procesador los recibe y debe saber reconocer si un determinado código, por ejemplo 1001101010011010100101101 10110101, forma parte de una instrucción de un programa, constituye parte de un texto, es una cifra numérica que se necesita para un cálculo o se trata del color de un pixel de una foto.

Para que no haya confusiones, es imprescindible establecer unas normas muy estrictas o **estándares** en el modo en que se componen los códigos binarios. Todas las letras deben tener la misma longitud, todos los píxeles de una foto utilizarán la misma longitud de código para fijar el color, etc.

También va a ser necesario etiquetar los bloques de datos con códigos que le permitan saber al ordenador, por ejemplo, dónde empieza un texto, con un código **Start of text** (carácter ASCII #02) y dónde termina, con un código **End of text** (carácter ASCII #03).

Y, para terminar, será necesario extremar las medidas de orden en el modo en que se almacenan los datos, utilizando códigos que etiqueten el contenido de los archivos, como las conocidas extensiones de archivo: **odt** (documento de texto en formato abierto), **jpg** (archivo gráfico comprimido) o **html** (documento de hipertexto para la web) por ejemplo.

Para saber más, conviene que consultes en Internet cualquiera de los glosarios de extensiones y formatos de archivo como este: <http://www.hispazone.com/conttuto.asp?IdTutorial=91>

Luis González
Profesor de Tecnologías de la Información
Departamento de Tecnología
I.E.S. Santa Eugenia (Madrid)